In [ ]:
```python
from functions import *
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt
plt.style.use(plt.style.available[5])
```

6.1. Color space

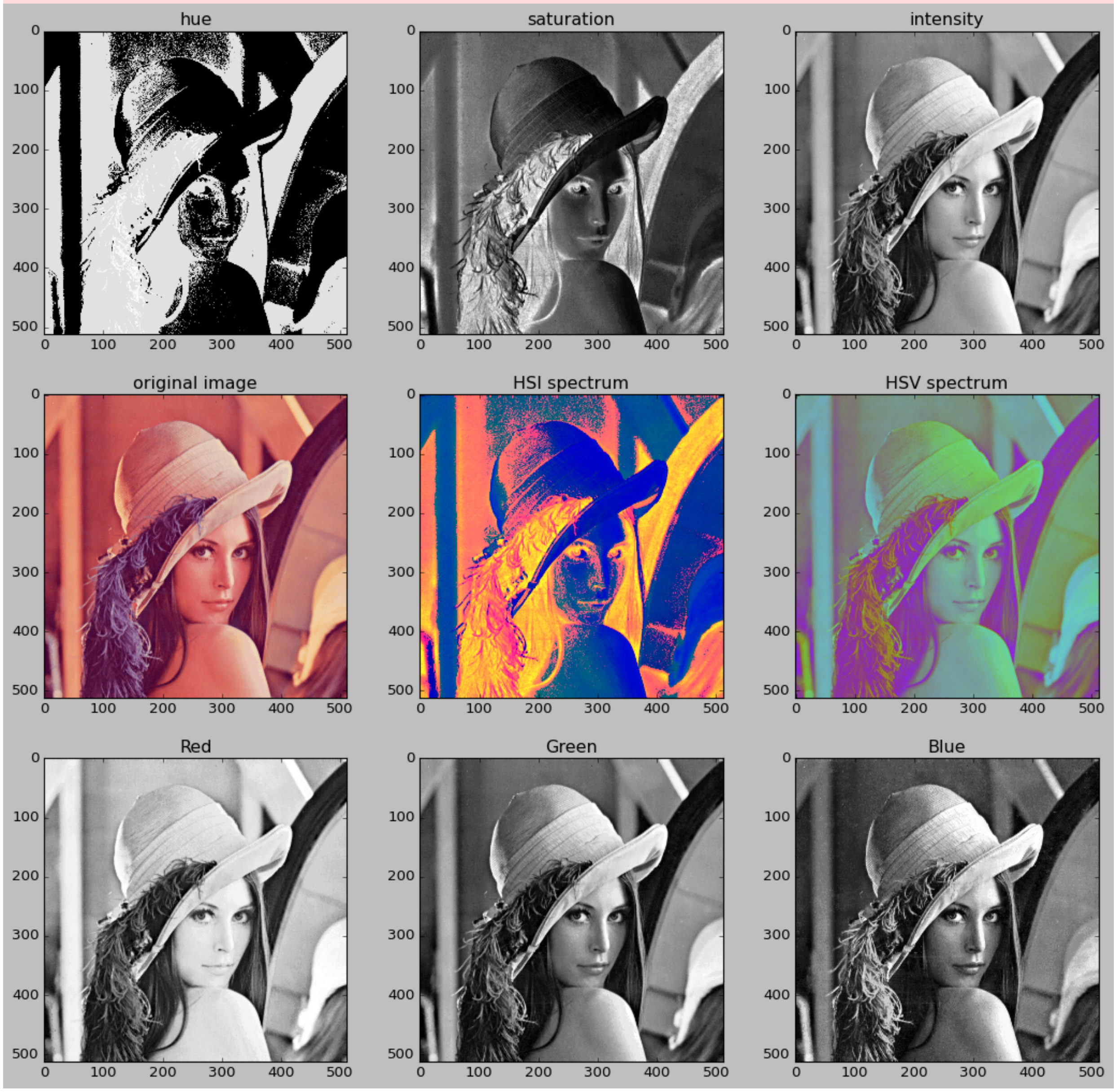
6.1.1. Convert Lena to HSI format, and display the HIS components as separate grayscale images. Observe these images to comment on what does each of the H, S, I components represent. The HSI images should be saved in double precision.

In [ ]:
```python
lena = cv2.imread('Lena.bmp')
lena = cv2.cvtColor(lena, cv2.COLOR_BGR2RGB)

# red,green,blue = cv2.split(lena)
# lena_abs = cv2.merge((red,green,blue))

lena_hsv = cv2.cvtColor(lena,cv2.COLOR_BGR2HLS)


h,s,i = RGB2HSI(lena)
h = normalize(h,(math.pi*2),0)
s = normalize(s,1,0)
# i = normalize(i,255,0)
lena_hsi = cv2.merge((h,s,i))
```

In [ ]:
```python
figure = plt.figure(figsize=(16,15))

figure.add_subplot(3,3,1)
plt.imshow(h,cmap='gray')
plt.title('hue',color='black')

figure.add_subplot(3,3,2)
plt.imshow(s,cmap='gray')
plt.title('saturation',color='black')

figure.add_subplot(3,3,3)
plt.imshow(i,cmap='gray')
plt.title('intensity',color='black')

figure.add_subplot(3,3,4)
ha,la,sa = cv2.split(lena)
plt.imshow(lena)
plt.title('original image',color='black')

figure.add_subplot(3,3,5)
plt.imshow(lena_hsi)
plt.title('HSI spectrum',color='black')

figure.add_subplot(3,3,6)
plt.imshow(lena_hsv)
plt.title('HSV spectrum',color='black')
```

```python
red,green,blue = cv2.split(lena)

figure.add_subplot(3,3,7)
plt.imshow(red,cmap='gray')
plt.title('Red',color='black')

figure.add_subplot(3,3,8)
plt.imshow(green,cmap='gray')
plt.title('Green',color='black')

figure.add_subplot(3,3,9)
plt.imshow(blue,cmap='gray')
plt.title('Blue',color='black')

plt.show()
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



6.1.2. *Present and discuss new color space (at least three) in detail which was not introduced in class (Application, Equation, etc.).

6.2. Quantization

In [ ]:
```
from functions import *
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt
plt.style.use(plt.style.available[5])
```

6.2.1. Implement uniform quantization of a color image. Your program should do the following:

1. Read a grayscale image into an array.
2. Quantize and save the quantized image in a different array.
3. Compute the MSE and PSNR between the original and quantized images.
4. Display and print the quantized image.

Notice, your program should assume the input values are in the range of (0,256), but allow you to vary the reconstruction level. Record the MSE and PSNR obtained with L = 64, 32, 16, 8 and display the quantized images with corresponding L values. Comment on the image quality as you vary L. (Test on Lena Image).

In [ ]:
```
lena = cv2.imread('Lena.bmp',cv2.IMREAD_GRAYSCALE)

lena64 = quantize(lena.copy(),6)
lena32 = quantize(lena.copy(),5)
lena16 = quantize(lena.copy(),4)
lena8 = quantize(lena.copy(),3)
```

In [ ]:
```
figure = plt.figure(figsize=(11,10))

figure.add_subplot(2,2,1)
plt.imshow(lena64,cmap='gray')
plt.title('lena64',color='black')

figure.add_subplot(2,2,2)
plt.imshow(lena32,cmap='gray')
plt.title('lena32',color='black')

figure.add_subplot(2,2,3)
plt.imshow(lena16,cmap='gray')
plt.title('lena16',color='black')

figure.add_subplot(2,2,4)
plt.imshow(lena8,cmap='gray')
plt.title('lena8',color='black')
```

Out[ ]:
```
Text(0.5, 1.0, 'lena8')
```

In [ ]:
```
print(f'L=64: PSNR= {PSNR(srcImage=lena,testImage=lena64)}, MSE= {mean_square_error(im
print(f'L=32: PSNR= {PSNR(srcImage=lena,testImage=lena32)}, MSE= {mean_square_error(im
print(f'L=16: PSNR= {PSNR(srcImage=lena,testImage=lena16)}, MSE= {mean_square_error(im
print(f'L=8: PSNR= {PSNR(srcImage=lena,testImage=lena8)}, MSE= {mean_square_error(imag
```

```
L=64: PSNR= 42.66330132120192, MSE= 3.521683
L=32: PSNR= 35.80317064962315, MSE= 17.090836
L=16: PSNR= 29.16343936072466, MSE= 78.838150
L=8: PSNR= 23.169236343419712, MSE= 313.441666
```

6.2.2. For the Lena image, quantize the R, G, and B components to 3, 3, and 2 bits, respectively, using a uniform quantizer. Display the original and quantized color image. Comment on the difference in color accuracy.

In [ ]:
```
lena = cv2.imread('Lena.bmp')
lena = cv2.cvtColor(lena, cv2.COLOR_BGR2RGB)
red,green,blue = cv2.split(lena)

newRed = quantize(red,3)
newGreen = quantize(green,3)
newBlue = quantize(blue,2)
```

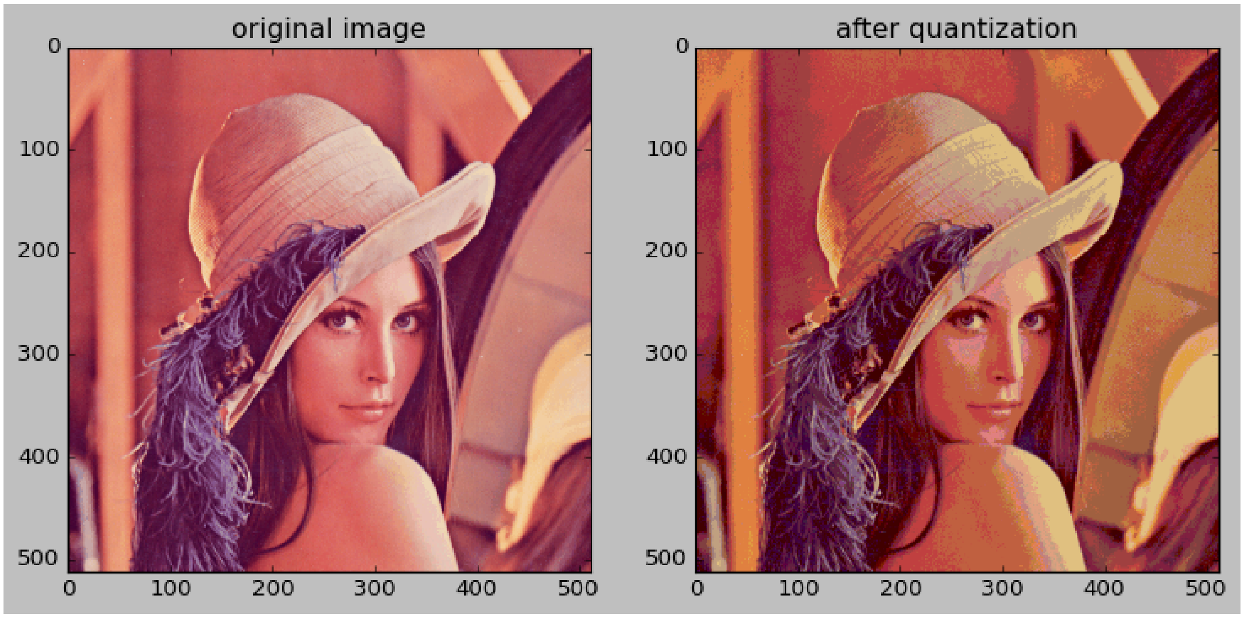
```
lena_quantized = cv2.merge((newRed,newGreen,newBlue))
```

In [ ]:

```
figure = plt.figure(figsize=(11,10))

figure.add_subplot(1,2,1)
plt.imshow(lena)
plt.title('original image',color='black')

figure.add_subplot(1,2,2)
plt.imshow(lena_quantized)
plt.title('after quantization',color='black')
```

Out[ ]:

```
Text(0.5, 1.0, 'after quantization')
```



6.2.3. We want to weave the Baboon image on a rug. To do so, we need to reduce the number of colors in the image with minimal visual quality loss. If we can have 32, 16 and 8 different colors in the weaving process, reduce the color of the image to these three special modes. Discuss and display the results.

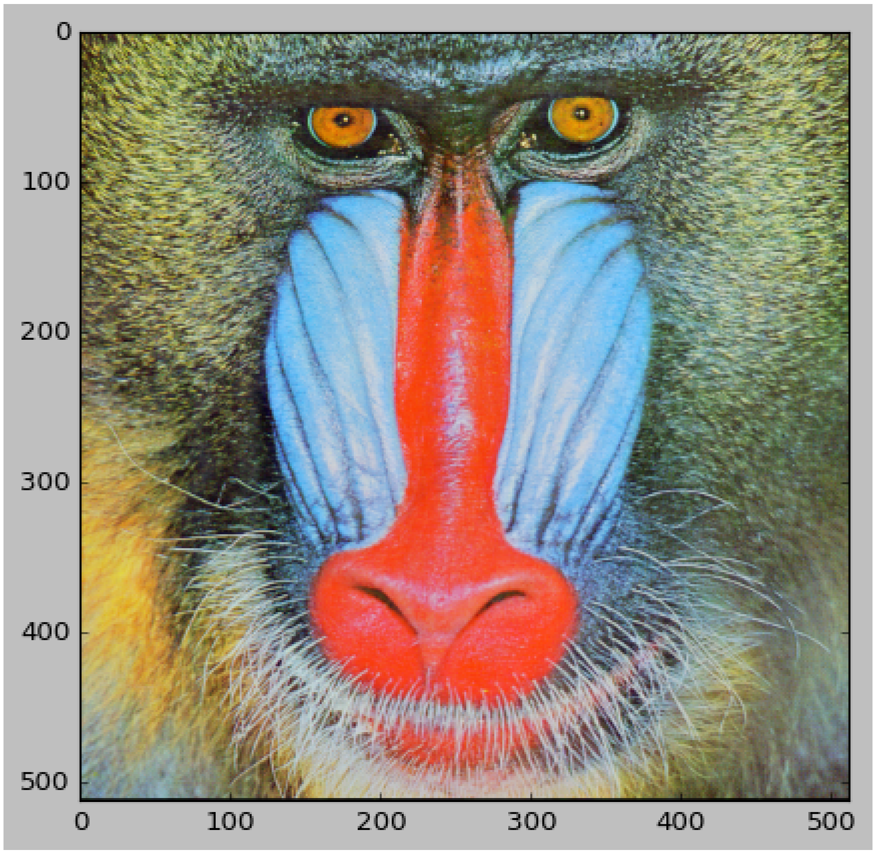
Note: you can use immse and psnr for problem 6.2.

In [ ]:

```
baboon = cv2.imread('baboon.bmp')
baboon = cv2.cvtColor(baboon, cv2.COLOR_BGR2RGB)
red,green,blue = cv2.split(baboon)

plt.imshow(baboon)
```

Out[ ]:

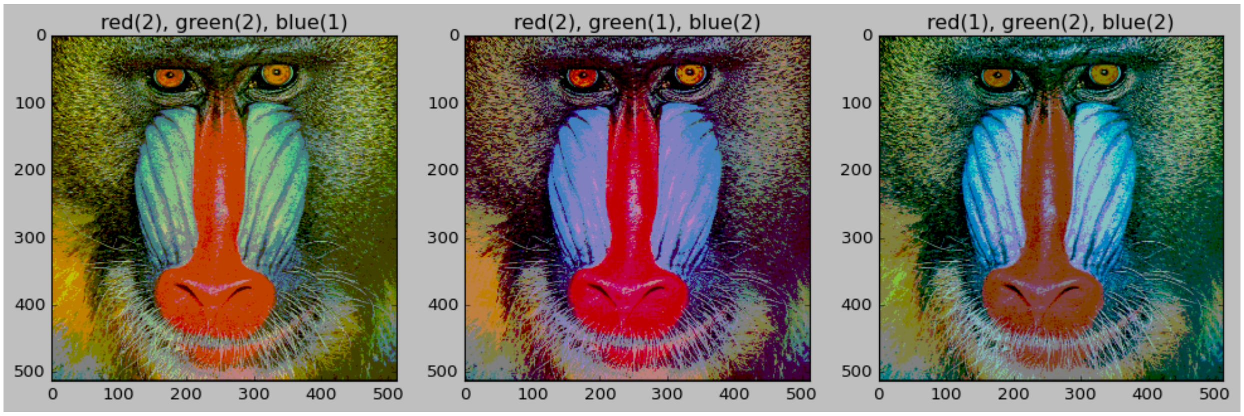
```
<matplotlib.image.AxesImage at 0x1ca9df61270>
```



In [ ]:

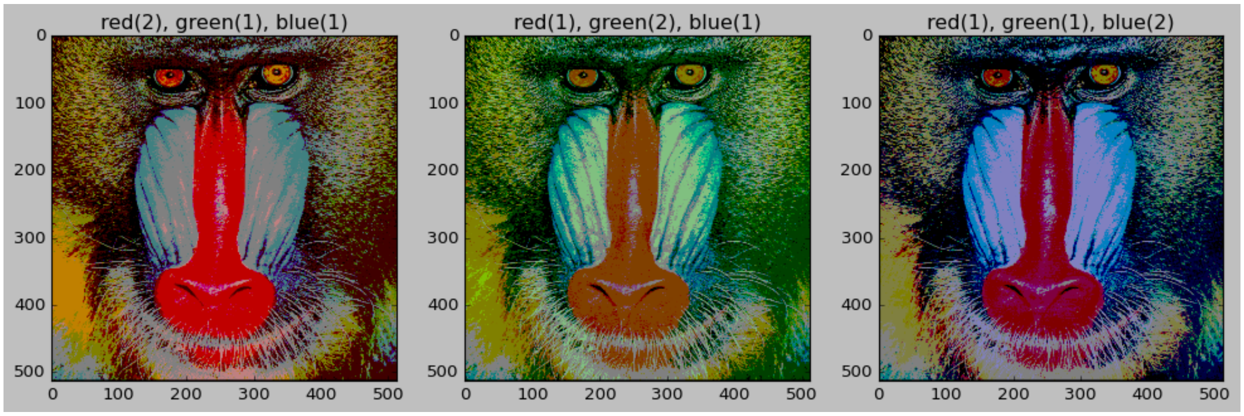
```
figure = plt.figure(figsize=(15,15))

# for 32 colors
newRed = quantize(red,2)
newGreen = quantize(green,2)
newBlue = quantize(blue,1)

baboon32colors = cv2.merge((newRed,newGreen,newBlue))

figure.add_subplot(1,3,1)
plt.imshow(baboon32colors)
plt.title('red(2), green(2), blue(1)',color='black')


newRed = quantize(red,2)
newGreen = quantize(green,1)
newBlue = quantize(blue,2)

baboon32colors = cv2.merge((newRed,newGreen,newBlue))

figure.add_subplot(1,3,2)
plt.imshow(baboon32colors)
plt.title('red(2), green(1), blue(2)',color='black')

newRed = quantize(red,1)
newGreen = quantize(green,2)
newBlue = quantize(blue,2)

baboon32colors = cv2.merge((newRed,newGreen,newBlue))

figure.add_subplot(1,3,3)
plt.imshow(baboon32colors)
plt.title('red(1), green(2), blue(2)',color='black')
```

```python
plt.show()
```



In [ ]:

```python
figure = plt.figure(figsize=(15,15))

# for 16 colors
newRed = quantize(red,2)
newGreen = quantize(green,1)
newBlue = quantize(blue,1)

baboon16colors = cv2.merge((newRed,newGreen,newBlue))

figure.add_subplot(1,3,1)
plt.imshow(baboon16colors)
plt.title('red(2), green(1), blue(1)',color='black')


newRed = quantize(red,1)
newGreen = quantize(green,2)
newBlue = quantize(blue,1)

baboon16colors = cv2.merge((newRed,newGreen,newBlue))

figure.add_subplot(1,3,2)
plt.imshow(baboon16colors)
plt.title('red(1), green(2), blue(1)',color='black')

newRed = quantize(red,1)
newGreen = quantize(green,1)
newBlue = quantize(blue,2)

baboon16colors = cv2.merge((newRed,newGreen,newBlue))

figure.add_subplot(1,3,3)
plt.imshow(baboon16colors)
plt.title('red(1), green(1), blue(2)',color='black')

plt.show()
```



In [ ]:

```
# for 8 colors
newRed = quantize(red,1)
newGreen = quantize(green,1)
newBlue = quantize(blue,1)

baboon8colors = cv2.merge((newRed,newGreen,newBlue))

plt.imshow(baboon8colors)
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x1ca9e3bd6f0>
```